AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims

Claim 1 (Previously Presented): A method of scanning a computer file comprising program code for virus infections, the method comprising:

a) identifying, from a set of predetermined compilers, a compiler used to create the program code in the file;

b) determining a frequency distribution of selected machine code instructions or sequences of such instructions in the program code in the file;

c) maintaining a database holding frequency distributions of machine code instructions or sequences thereof expected for respective compilers in the set of predetermined compilers; and

d) flagging the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with the frequency distribution of machine code instructions or sequences thereof stored in the database as being expected for the compiler identified in step a).

Claim 2 (Previously Presented): A method according to claim 1 wherein step b) comprises a step, working from an entry point of the program code, of

b1) tracing an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

Claim 3 (Previously Presented): A method according to claim 2 wherein when, during step b1), a subroutine call or conditional branch instruction is encountered, a destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with following instructions, if any.

Claim 4 (Previously Presented): A method according to claim 1 wherein the program code is examined for opcode constructs, which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing.

Claim 5 (Previously Presented): A method according to claim 1, further comprising a step, where step d) flags the file as possibly viral, of comparing the program code with a list of permissible exceptions and suppressing the flag if the program code is considered to be in the exception list.

Claim 6 (Currently Amended): A ~~computer~~ system implemented on a computer apparatus and operative to scan a computer file comprising program code for virus infections, the system comprising:

a compiler analyser operative to identify, from a set of predetermined compilers, a compiler used to create the program code in the file;

an instruction frequency analyser operative to determine a frequency distribution of selected machine code instructions or sequences of such instructions in the program code in the file;

a database holding frequency distributions of machine code instructions or sequences thereof expected for respective compilers in the set of predetermined compilers; and

a frequency distribution checker operative to flag the file as possibly infected with a virus, or not, on the basis of comparison of the determined frequency distribution with the frequency distribution of machine code instructions or sequences thereof stored in the database as being expected for the compiler identified by the compiler analyser.

Claim 7 (Previously Presented): A system according to claim 6, wherein the instruction frequency analyser is operative working from an entry point of the program code, to trace an execution graph by decoding successive instruction opcodes and updating frequency counts of decoded instructions as this tracing proceeds.

Claim 8 (Previously Presented): A system according to claim 7, wherein the instruction frequency analyser is operable such that when a subroutine call or conditional branch instruction is encountered, a destination of the call or branch instruction is pushed onto a stack, tracing proceeds into the subroutine call, and when a return instruction is encountered, the pushed location is popped from the stack and tracing continues with following instructions, if any.

Claim 9 (Previously Presented): A system according to claim 6, wherein the frequency distribution checker is operative to examine the program code for opcode constructs, which are expected to occur a known ratio to each other and, if the ratio actually found differs from the known one by more than a certain amount, the file is flagged as possibly viral, or subject to further processing.

Claim 10 (Previously Presented): A system according to claim 6, further comprising an exception list checker operative, when the frequency distribution checker flags the file as possibly viral, to compare the program code with a list of permissible exceptions and to suppress the flag if the program code is considered to be in the exception list.

Claim 11 (Previously Presented): A method according to claim 4, wherein the opcode constructs include subroutine-call and sub-routine return instruction sequences.

Claim 12 (Previously Presented): A system according to claim 9, wherein the opcode constructs include subroutine-call and subroutine-return instruction sequences.

Claims 13 and 14 (Canceled).